EDO UNIVERSITY IYAMHO

Department of Computer Science

CMP 211: Software Development Practice I

Instructor: *Mr. Acheme David* email: acheme.david@edouniversity.edu.ng

Lectures: Mondays, 1pm – 3pm, SH1, phone: (+234) 802889197

Office hours: Mon-Fri 8:00am to 4 PM (except lectures hours).

Description: This course is intended to give the students a deeper understanding of software development and Object Oriented Programming. The C++ programming Language is used to illustrate this.

Prerequisites: CMP 121: Structural Programming Using FORTRAN

Assignments: We expect to have 5 individual homework assignments throughout the course in addition to a Mid-Term Test and a Final Exam. Home works are due at the beginning of the class on the due date. Home works are organized and structured as preparation for the midterm and final exam, and are meant to be a studying material for both exams.

Grading: We will assign 15% of this class grade to homework/assignments, 15% for the mid-term test, and 70% for the final exam. The Final exam is comprehensive.

Lectures: Below is a description of the contents. We may change the order to accommodate the materials you need for the projects.

1 Introduction

Visual C++ is a part of Microsoft Visual Studio 2010 for developing programs in C++ programming language. This first lectures introduces how to create a project, create a program, compile and run the program in Visual C++. The screen shots are from Visual Studio 2010, but its all same for later versions up to Studio 2017

2 Getting Started with Visual C++

Visual C++ is easy to install. Visit the official Website for free download. Suppose you have installed Visual C++ 2010 Express Edition. You can launch Visual C++ from the Windows Start button by choosing *Visual Studio 2010.* The Visual C++ 2010 the user interface appears, as shown in Figure 1.

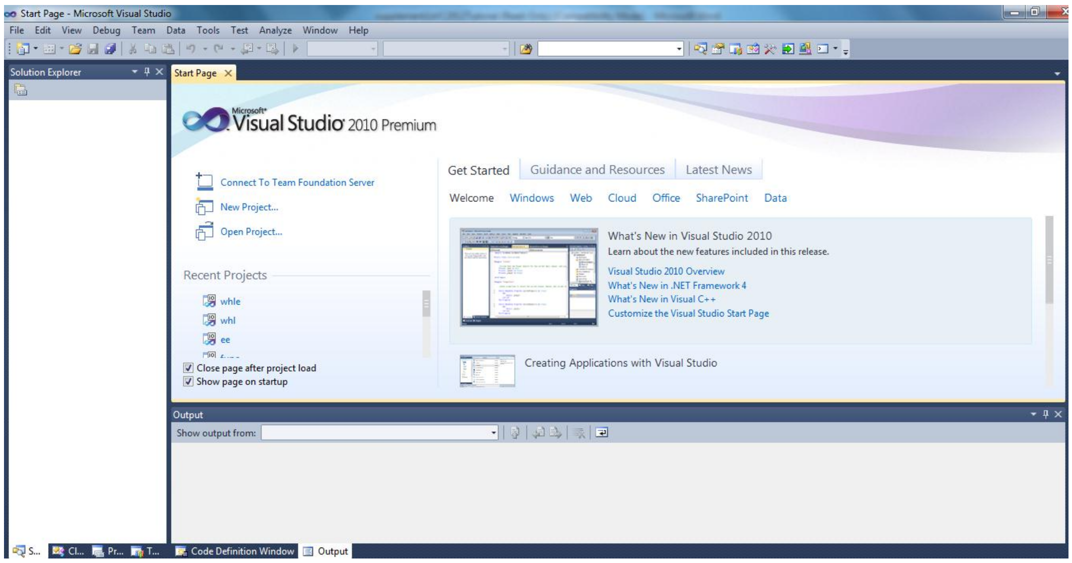

Start Page - Microsoft Visual Studio
File Edit View Debug Team Data Tools Test Analyze Window Help
Solution Explorer
Start Page
Microsoft Visual Studio 2010 Premium
Connect To Team Foundation Server
New Project...
Open Project...
Recent Projects
whle
whl
ee
Close page after project load
Show page on startup
Get Started
Guidance and Resources
Latest News
Welcome Windows Web Cloud Office SharePoint Data
What's New in Visual Studio 2010
Learn about the new features included in this release.
Visual Studio 2010 Overview
What's New in .NET Framework 4
What's New in Visual C++
Customize the Visual Studio Start Page
Creating Applications with Visual Studio
Output
Show output from:
Code Definition Window
Output

Figure 1 *The Visual C++ user interface is a single window that performs editing, compiling, debugging, and running programs.*

3 Creating a Project

To create C++ programs in Visual C++, you have to first create a project. A project is like a holder (folder) that ties all the files together. Here are the steps to create a project:

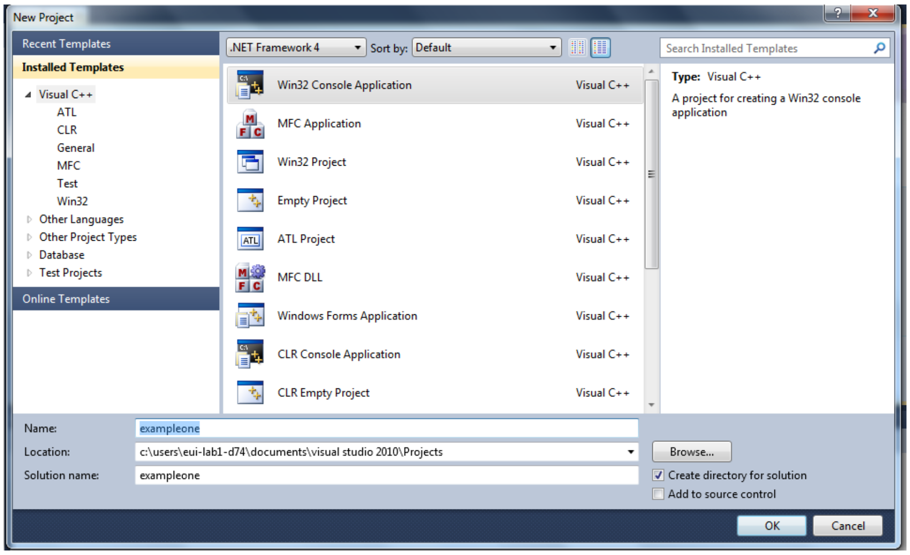
Choose **File**, **New Project** to display the New Project window, as shown in Figure 2.

1. Choose *C++* under the Template and select **Win32 Console Application** in the middle column. Type <u>exampleone</u> in the Name field and leave

it at the default location. Click *OK* to display the Win32 Application Wizard window, as shown in Figure 3.

2. Click **Next** to display the application settings window, as shown in Figure 4.

3. Select **Console application** in the Application type section and check **Empty project** in the Additional options section. Click *Finish* to create a project. You will see the project named exampleone in the Solution Explorer, as shown in Figure 5.



Win32 Console Application	Visual C++
MFC Application	Visual C++
Win32 Project	Visual C++
Empty Project	Visual C++
ATL Project	Visual C++
MFC DLL	Visual C++
Windows Forms Application	Visual C++
CLR Console Application	Visual C++
CLR Empty Project	Visual C++

Type: Visual C++

A project for creating a Win32 console application

Name: exampleone

Location: c:\users\eui-lab1-d74\documents\visual studio 2010\Projects

Solution name: exampleone

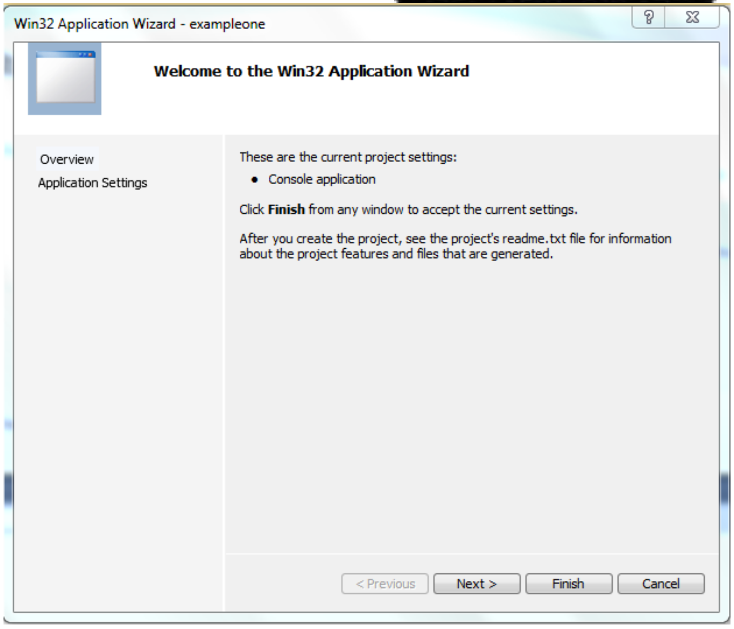
Browse...

[✓] Create directory for solution

[ ] Add to source control

OK Cancel

Figure 2 *You need to create a project before creating programs.*



Win32 Application Wizard - exampleone

Welcome to the Win32 Application Wizard

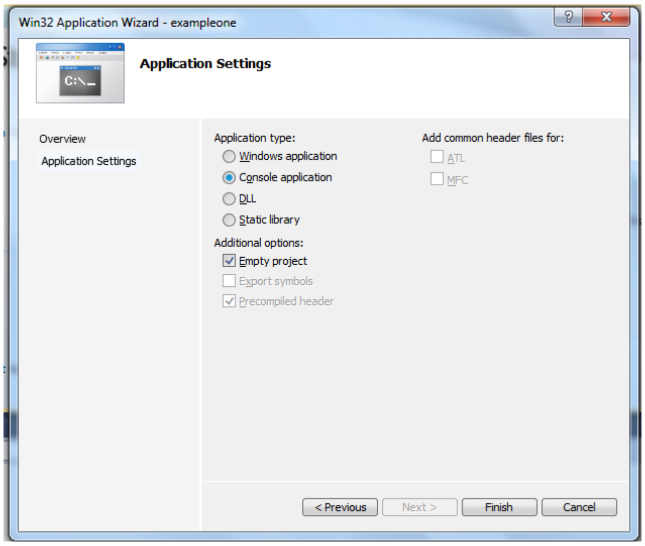
Overview
Application Settings

These are the current project settings:
- Console application

Click **Finish** from any window to accept the current settings.

After you create the project, see the project's readme.txt file for information about the project features and files that are generated.

Figure 3 *Win32 Application Wizard creates a project for Win32 applications.*



Win32 Application Wizard - exampleone

Application Settings

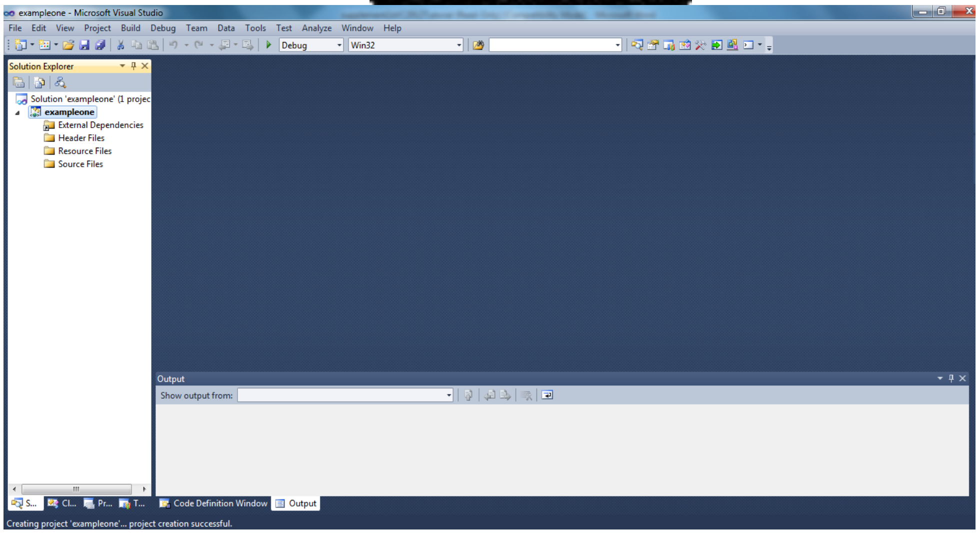
Overview
Application Settings

Application type:
- ○ Windows application
- ● Console application
- ○ DLL
- ○ Static library

Additional options:
- ☑ Empty project
- ☐ Export symbols
- ☑ Precompiled header

Add common header files for:
- ☐ ATL
- ☐ MFC

Win32 Application Settings window lets you set the application type.



Creating project 'exampleone'... project creation successful.

A project is created for C++ console applications.

4 Creating a C++ Program

After you create a project, you can create programs in it. Here are the steps to create a C++ program for Listing 1.1:

1. Right-click the exampleone project in the Solution Explorer to display a context menu. Choose **Add**, **Add NewItem** from the context menu of the exampleone project (see Figure 6) to display the Add New Item window, as shown in Figure 7.
2. Choose Code under Visual C++ on the left column and C++ File (.cpp) in the middle column. Enter <u>Welcome</u> in the Name field. Click **Add** to create the file, as shown in Figure 8.
3. Enter the code for Welcome.cpp exactly from Listing 1.1, as shown in Figure 9.

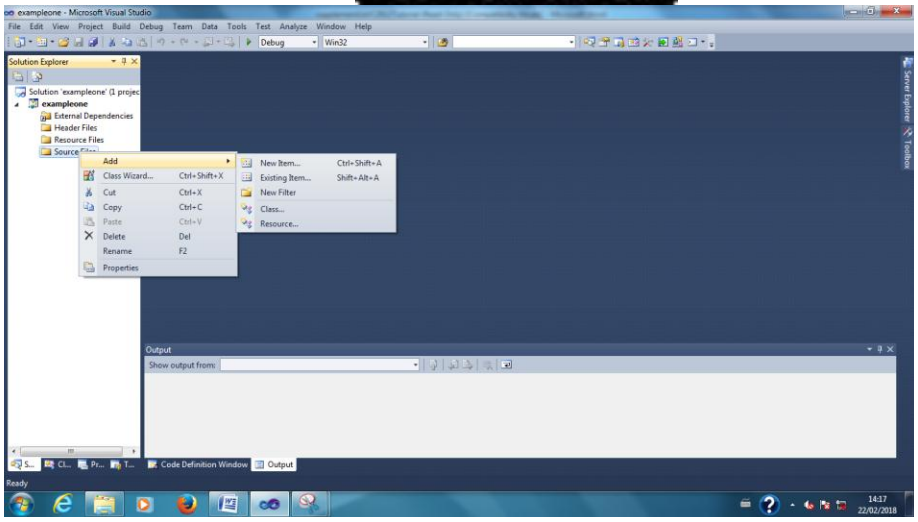
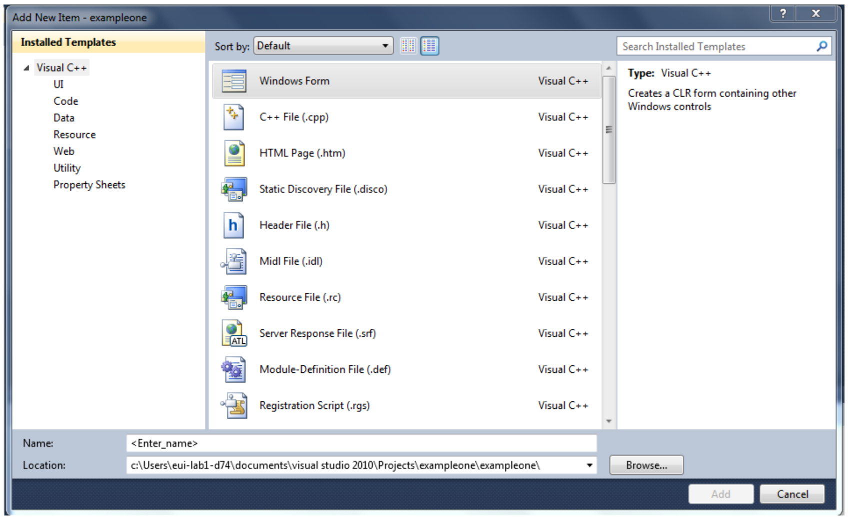


Figure 6



Sort by: Default

Windows Form	Visual C++
C++ File (.cpp)	Visual C++
HTML Page (.htm)	Visual C++
Static Discovery File (.disco)	Visual C++
Header File (.h)	Visual C++
Midl File (.idl)	Visual C++
Resource File (.rc)	Visual C++
Server Response File (.srf)	Visual C++
Module-Definition File (.def)	Visual C++
Registration Script (.rgs)	Visual C++

Type: Visual C++

Creates a CLR form containing other Windows controls

Name: <Enter_name>

Location: c:\Users\eui-lab1-d74\documents\visual studio 2010\Projects\exampleone\exampleone\

Figure 7

You can open the Add New Item window from the project's context menu. You can specify the file type, name, and location to create a file.

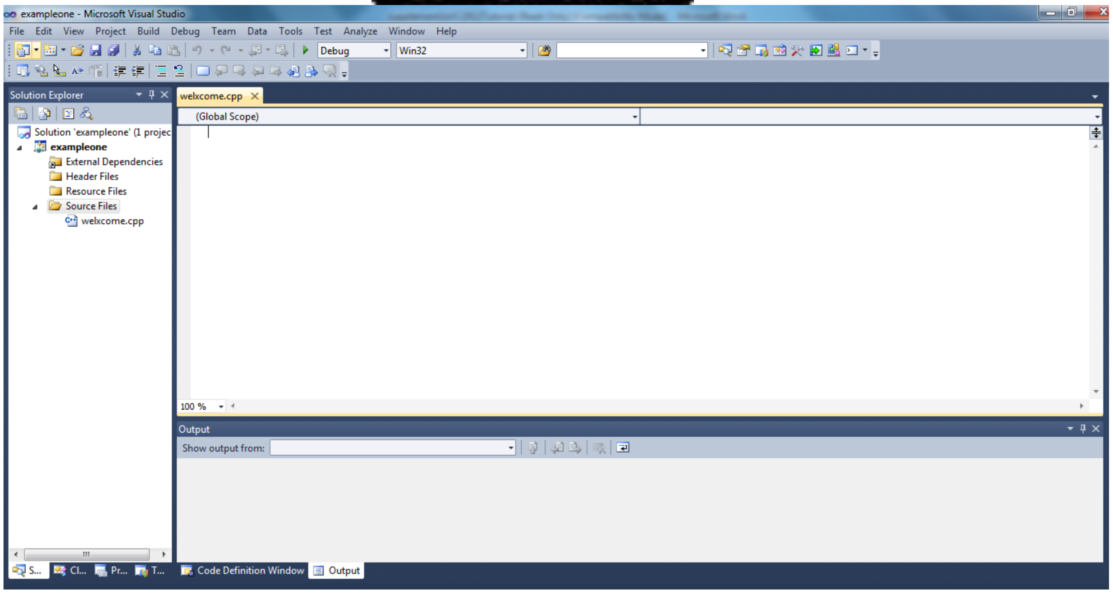
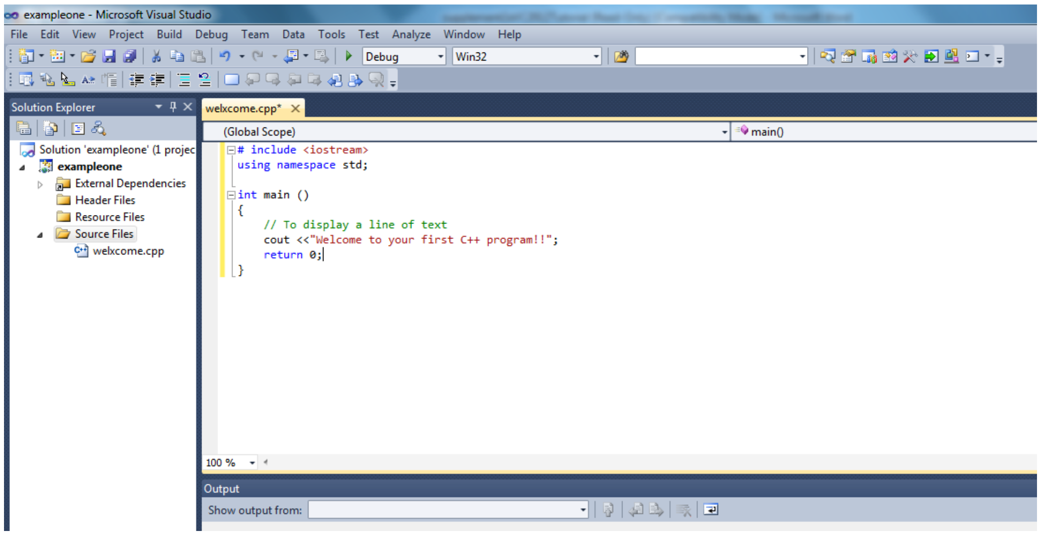


Figure 8 *Welcome.cpp is created in the project.*

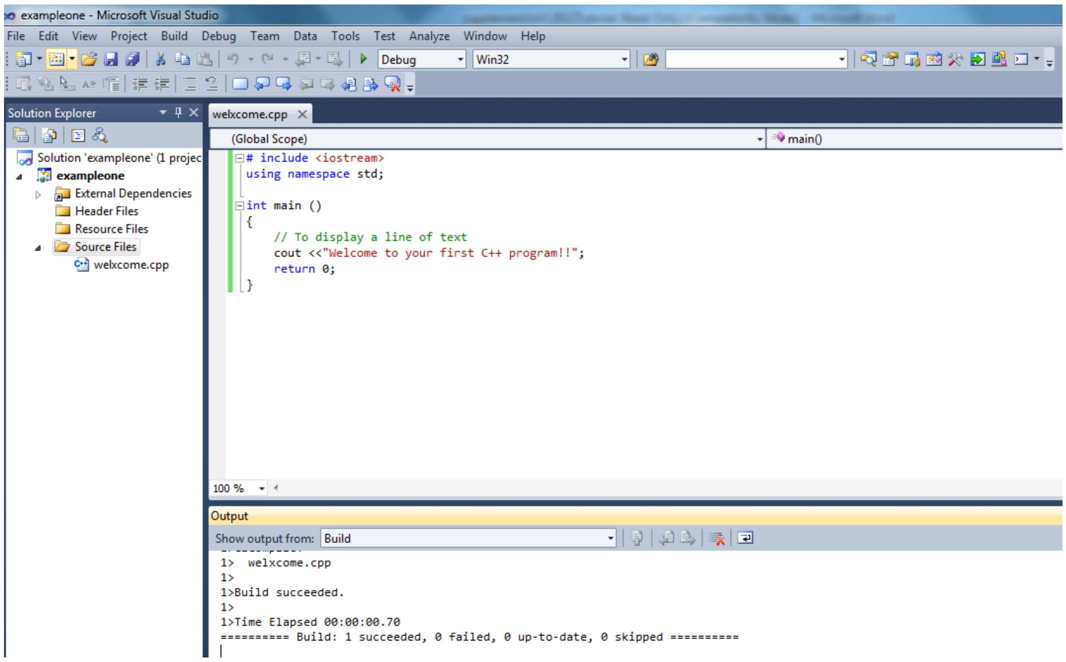


```
# include <iostream>
using namespace std;

int main ()
{
    // To display a line of text
    cout <<"Welcome to your first C++ program!!";
    return 0;
}
```

Figure 9: *The source code for Welcome.cpp is entered.*

5 Compiling a C++ Program

After you create a program, you can compile it. You may do so by choosing *Build*, *Compile*, or press *Ctrl+F7*, or choose *Compile* in the context menu for Welcome.cpp, as shown in Figure 10.



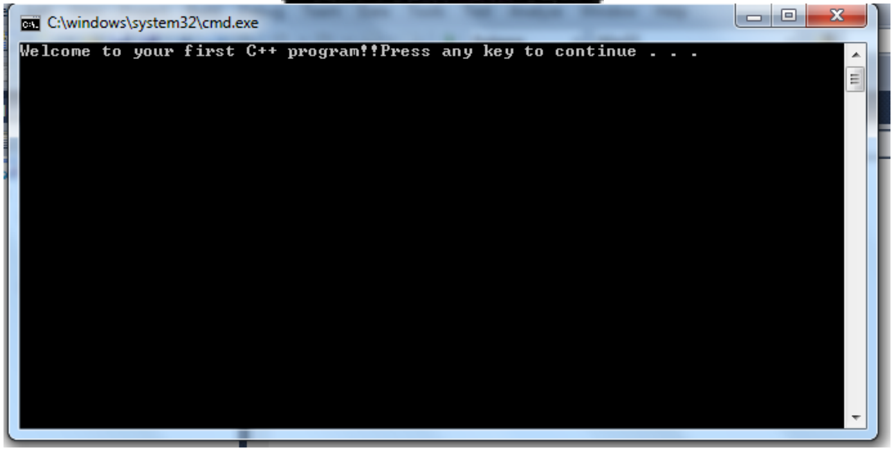
```
# include <iostream>
using namespace std;

int main ()
{
    // To display a line of text
    cout <<"Welcome to your first C++ program!!";
    return 0;
}
```

```
1>  welxcome.cpp
1>
1>Build succeeded.
1>
1>Time Elapsed 00:00:00.70
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

Figure 10 *Choose the Compile command to compile the program.*

6 Running a C++ Program

To run the program, press *Ctrl+F5*. You will see the output displayed in a DOS window, as shown in Figure 11.



```
Welcome to your first C++ program!!Press any key to continue . . .
```

Figure 11 ***The output is displayed in a DOS window.***

YOUR FIRST C++ PROGRAM

```
1. #include <iostream.h>
2. int main()
3. {
4. cout<< "Hello World!\n";
5. return 0;
6. }
```

THE PARTS OF THE PROGRAM

On line 1, the file iostream.h is included in the file. The first character is the # symbol, which is a signal to the preprocessor. Each time you start your compiler, the preprocessor is run. The preprocessor reads through your source code, looking for lines that begin with this symbol (#), and acts on those lines before the compiler runs.

The file iostream.h (Input-Output-Stream) is used by cout, which assists with writing to the screen. The effect of line 1 is to include the file iostream.h into this program as if you had typed it in yourself.

Line 2 begins the actual program with a function named main(). Every C++ program has a main() function. In general, a function is a block of code that performs one or more actions. Usually functions are invoked or called by other functions, but main() is special. When your program starts, main() is called automatically.

All functions begin with an opening brace ({) and end with a closing brace (}). The braces for the main() function are on lines 4 and 7. Everything between the opening and closing braces is considered a part of the function.

The meat and potatoes of this program is on line 4. The object cout is used to print a message to the screen. We'll cover objects in generally later in this course when we treat Classes, and cout and its related object cin. These two objects, cout and cin, are used in C++ to print strings and values to the screen. A string is just a set of characters.

Here's how cout is used: type the word cout, followed by the output redirection operator (<<). Whatever follows the output redirection operator is written to the screen. If you want a string of characters written, be sure to enclose them in double quotes ("), as shown on line 5.

The final two characters, \n, tell cout to put a new line after the words Hello World!

All ANSI-compliant programs declare main() to return an int. This value is "returned" to the operating system when your program completes. Some programmers signal an error by returning the value 1. In this course, main() will always return 0. The main() function ends on line 6 with the closing brace.

USING COUT

To print a value to the screen, write the word cout, followed by the insertion operator (<<), which you create by typing the less-than character (<) twice. Even though this is two characters, C++ treats it as one.

Follow the insertion character with your data. The next example illustrates this:

Write this example exactly as written, except substitute your own name where you see Jesse
Liberty.

```

#include <iostream.h>
intmain()
{
cout<< "Hello there.\n";
cout<< "Here is 5: " << 5 << "\n";
cout<< "The manipulator endl writes a new line to the screen." ;
cout<< "Here is a very big number:\t" << 70000 <<endl;
cout<< "Here is the sum of 8 and 5:\t" << 8+5 <<endl;
cout<< "Here's a fraction:\t\t" << (float) 5/8 <<endl;
cout<< "And a very very big number:\t" << (double) 7000
* 7000 <<endl;
cout<< "Don't forget to replace Jesse Liberty with your
name...\n";
cout<< "Jesse Liberty is a C++ programmer!\n";
return 0;
}
```

FUNCTIONS

A function is, in effect, a subprogram that can act on data and return a value. Every C++ program has at least one function, main(). When your program starts, main() is called automatically. main() might call other functions, some of which might call still others.

Each function has its own name, and when that name is encountered, the execution of the program branches to the body of that function. When the function returns, execution resumes on the next line of the calling function.

When a program calls a function, execution switches to the function and then resumes at the line after the function call. Well-designed functions perform a specific and easily understood task.

Complicated tasks should be broken down into multiple functions, and then each can be called in turn. Functions come in two varieties: userdefined and built-in. Built-in functions are part of your compiler package--they are supplied by the manufacturer for your use

Declaring and Defining Functions

Using functions in your program requires that you first declare the function and that you then define

the function. The declaration tells the compiler the name, return type, and parameters of the function.

The definition tells the compiler how the function works. No function can be called from any other function that hasn't first been declared. The declaration of a function is called its prototype

Declaring the Function

There are three ways to declare a function:

Write your prototype into a file, and then use the #include directive to include it in your program.

Write the prototype into the file in which your function is used.

Define the function before it is called by any other function. When you do this, the definition acts as its own declaration.

Function Prototypes

Many of the built-in functions you use will have their function prototypes already written in the files you include in your program by using #include.

For functions you write yourself, you must include

the prototype. The function prototype is a statement, which means it ends with a semicolon. It consists of the function's return type, name, and parameter list.

The parameter list is a list of all the parameters and their types, separated by commas.

The function prototype and the function definition must agree exactly about the **return type**, **the name**, and the **parameter list**. If they do not agree, you will get a compile-time error.

Note, however, that the function prototype does not need to contain the names of the parameters, just their types. A prototype that looks like this is perfectly legal:

```
long Area(int, int);
```

This prototype declares a function named Area() that returns a long and that has two parameters, both integers. Although this is legal, it is not a good idea. Adding parameter names makes your prototype clearer. The same function with named parameters might be

```
long Area(int length, int width);
```

It is now obvious what this function does and what the parameters are. Note that all functions have a return type. If none is explicitly stated, the return type defaults to int.

Your programs will be easier to understand, however, if you explicitly declare the return type of every function, including main(). Listing 5.1 demonstrates a program that includes a function prototype for the Area() function.

Function Prototype Syntax

```
return_typefunction_name( [type
[parameterName]]...);
```

Function Definition Syntax

```
return_typefunction_name( [type parameterName]...)
{

}
```

AN EXAMPLE

```
#include <iostream>
using namespace std;
intFindArea(int length, int width); //function prototype
int main()
{
intlengthOfYard;
intwidthOfYard;
intareaOfYard;
cout<< "How wide is your yard? "<<endl; •  cin>>widthOfYard;
cout<< "How long is your yard? " <<endl;
cin>>lengthOfYard;
areaOfYard= FindArea(lengthOfYard,widthOfYard);
cout<< "Your yard is\n";
cout<<areaOfYard;
cout<< " square feet\n\n";
return 0;
}
intFindArea(int l, int w)
{
```

```
return l * w;
}
```